Timing configuration on imperix controllers

PN259 | Posted on February 6, 2026 | Updated on April 10, 2026

Antonin STAMPBACH
Development Engineer
imperix

Table of Contents

- Timing architecture
- Timing configuration
- Basic timing configurations
  - Single rate update
  - Double rate update
- Advanced timing configurations
  - Postscaler
  - Data history
  - Variable switching frequency
  - Multi-rate execution
- Related topics

This article details the underlying clock architecture and timing configurations of imperix controllers, focusing on the four internal time bases (CLK0–CLK3) that govern ADC sampling, control loop execution, and PWM modulation. These timing principles are common across the entire imperix hardware family and are applicable to both standalone systems and synchronized multi-controller networks.
While the Simulation Essentials pages cover general model setup and Sampling Techniques explores data acquisition theory, this page provides the specific technical foundation for the clocking mechanisms that support both.

Additionally, a set of ready-to-use Simulink/PLECS example projects is provided to demonstrate the implementation of a double-rate update configuration.

Timing architecture

The timing architecture of imperix controllers typically differs from the distributed peripheral model of conventional microcontrollers by using a centralized FPGA-based timing engine. This approach abstracts the underlying hardware complexity, allowing the user to focus on control algorithms rather than taxing manual register management.

Conventional MCUs also often introduce timing jitter due to variable interrupt latencies (where the CPU's response time to an ADC trigger fluctuates). In contrast, the imperix architecture utilizes a 250 MHz global clock to align all resources within a single time domain. This design achieves nanosecond-level determinism between resources and bypasses the interrupt latencies found in traditional architectures.

The system also supports multiple phase-locked timing domains, maintaining precise synchronization across different frequencies in complex timing configuration. To accommodate various hardware requirements, the 250 MHz clock is divided into four distinct clocks, CLK0 through CLK3 (see Fig. 1), which can then be allocated to resources such as ADCs, PWM modulators, and CPU interrupts.

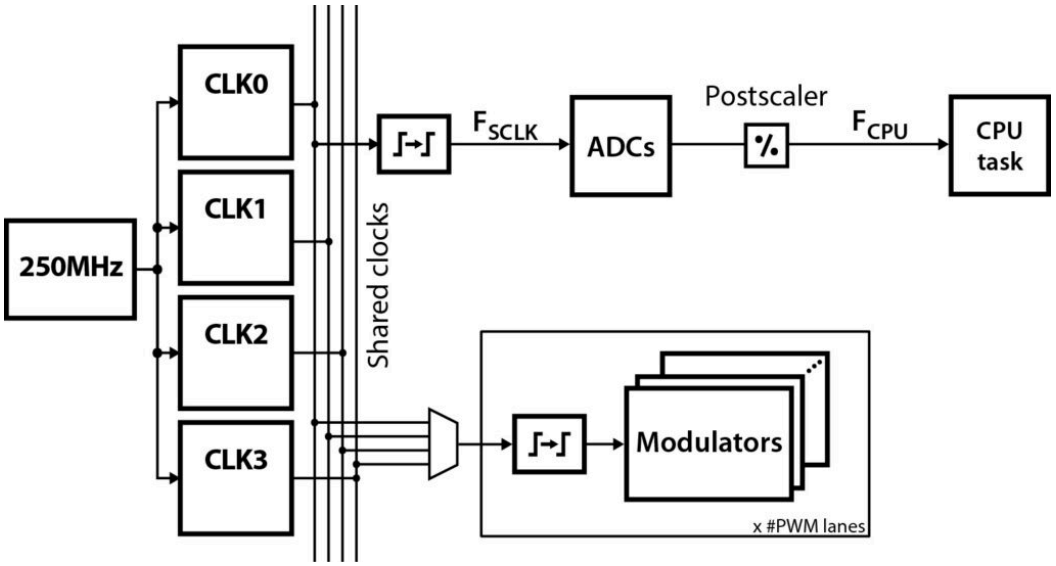

Figure 1 : Clocks generation in imperix controllers

Among the four time bases, CLK0 has a special role: it is always active and sets the timing for both sampling and CPU control execution. Two important internal signals are derived from CLK0, which are :

- The CPU interrupt clock (frequency F_{CPU}), which triggers the execution of the control task. When required, F_{CPU} can be obtained by decimating F_{SCLK} using a postscaler.
- SCLK, the physical sampling clock (frequency F_{SCLK}). It runs at the same rate as CLK0, but with a user-defined constant phase shift that sets the exact sampling instant.

Unlike CLK0, CLK1-3 can only be used as a time base for PWM modulators. A summary of the uses of the different clocks is given in Table 1.

Clock	Mandatory	Where to configure	Used for	Variable frequency support
CLK0	Yes	CONFIG block	CPU, ADC, PWM modulators	No
CLK1	No	CLK block	PWM modulators	Yes
CLK2	No	CLK block	PWM modulators	Yes
CLK3	No	CLK block	PWM modulators	Yes

Table 1 : Summary of clock functions and configuration

Finally, thanks to RealSync, all clocks within a network of imperix controllers are synchronized with a timing accuracy of ±2 ns. This means that the phase coherence and synchronization inherent to local resources, driven by the common 250 MHz time base, are maintained across the entire distributed network.

Timing configuration

At the software level, the four clocks described above are configured through the CONFIG and CLK blocks. These are available within the ACG SDK and illustrated in Fig. 2.

The CONFIG block is mandatory for every Simulink or PLECS model. Among its various functions, it handles the configuration of CLK0 and its associated signals. In contrast, CLK blocks (CLK1–3) are optional and can run at variable frequency. These clocks are only used to drive modulation blocks (PWMs), whenever the PWM time base must differ from CLK0. Further details regarding the simulation of these blocks can be found on the PN135 and PN137 pages.

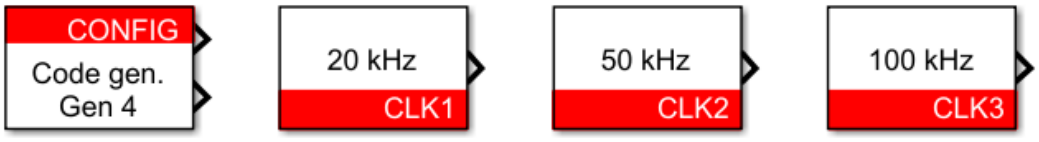

Figure 2 : Configuration blocks for the four different clocks (The displayed values are only for the sake of example)

In practice, most control schemes can rely on CLK0 only. CLK0 is configured in the CONFIG block and sets the sampling and CPU interrupt frequency, i.e., how often the control code is executed. In many applications, CLK0 is also used to define the PWM switching frequency. This common arrangement is often referred to as single rate update, and it covers a large fraction of standard use cases. This case is illustrated in Fig. 3.

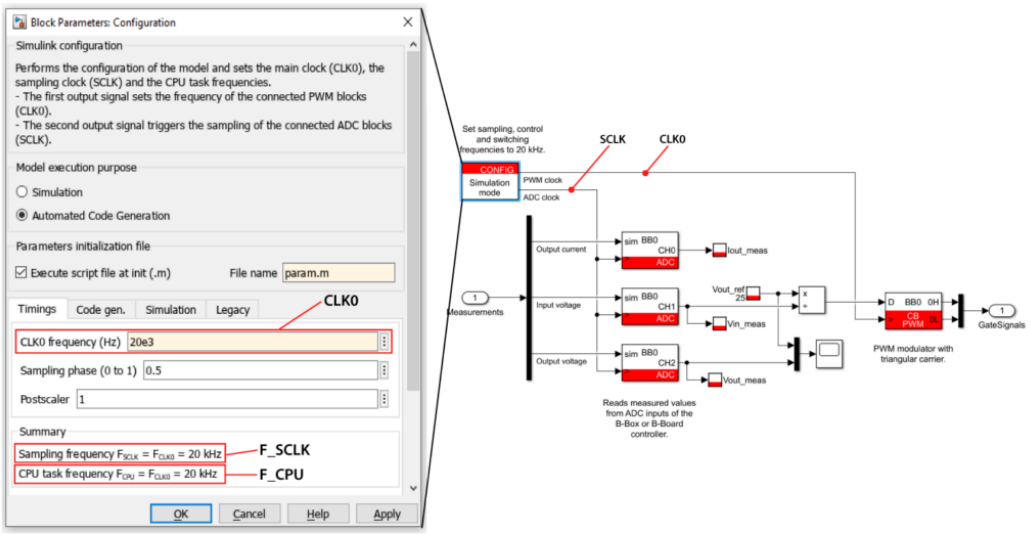

Figure 3 : Clocks setting and allocation in Simulink in single rate update mode configuration

Consequently, the clock configuration impacts three core subsystems as follows:

- **ADC** resources are always referenced to CLK0, which defines the sampling instants according to the physical sampling clock SCLK and with a frequency of F_{SCLK}.
- **CPU** execution is always tied to CLK0 with a configurable phase offset and postscaler.
- **PWM** blocks can be referenced to any of CLK0–CLK3. The selected PWM clock can be an integer multiple or submultiple of CLK0, or it may be non-integer related (including variable-frequency operation), depending on the modulation scheme and the chosen clock source.

Figure 4 shows the use of CLK1 in a configuration where the control loop is executed twice as fast as the PWMs. This configuration is also called double rate update and is detailed later.

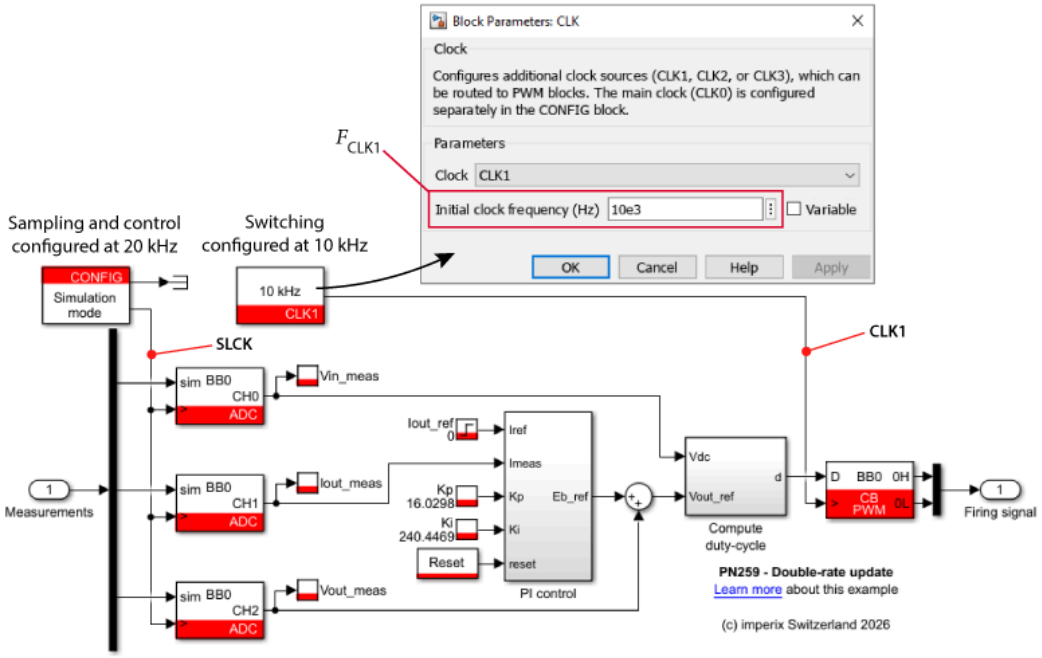

Figure 4 : Clocks setting and allocation in Simulink in double rate update configuration

Basic timing configurations

On imperix controllers, the timing configurations arise from the relationship between CLK0 and the selected PWM time base. While CLKO provides a fixed reference for ADC sampling and CPU execution, the PWM clocks determine the carrier frequency of the modulators and the specific instants at which the duty cycles are latched.

Those duty cycle update instants are not the same for all PWM carrier types. With a sawtooth (single-edge) carrier, there is only one effective update opportunity per PWM period, since the duty cycle is determined by a single intersection between the reference and the carrier. With a triangular (double-edge) carrier, two crossings occur per period, creating two distinct opportunities to update the duty cycle, as shown in Fig. 5.

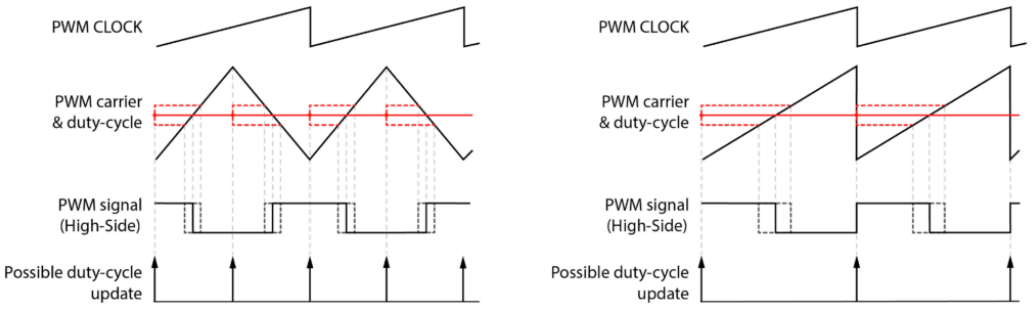

Figure 5 : PWM carriers and possible duty cycle update instant

Therefore, two fundamental timing configurations stem from these situations:

- Single rate update, in which case the control task is executed **once** per PWM period, and duty cycles are updated once per PWM period.
- Double rate update, in which case the control task is executed **twice** per PWM period, and duty cycles are updated twice per PWM period.

With the double-rate update configuration, the control performance is enhanced by doubling the execution frequency (F_{CPU}) relative to the PWM switching frequency (F_{SW}). By executing the control task twice per carrier period, the system achieves a higher effective sampling rate and more frequent duty cycle updates. This approach allows for a higher closed-loop bandwidth at a given switching frequency, improving the overall dynamic response without increasing switching losses.

Single rate update

Single rate update is the simplest timing configuration and remains the most widely used, as it covers a broad range of practical applications.

In a single rate configuration, ADC sampling, CPU control execution, and PWM duty cycle updating all occur at the same fundamental frequency (which is also the switching frequency). Consequently, only CLK0 is required, and it serves as the common time base. This configuration is represented in Fig. 6.

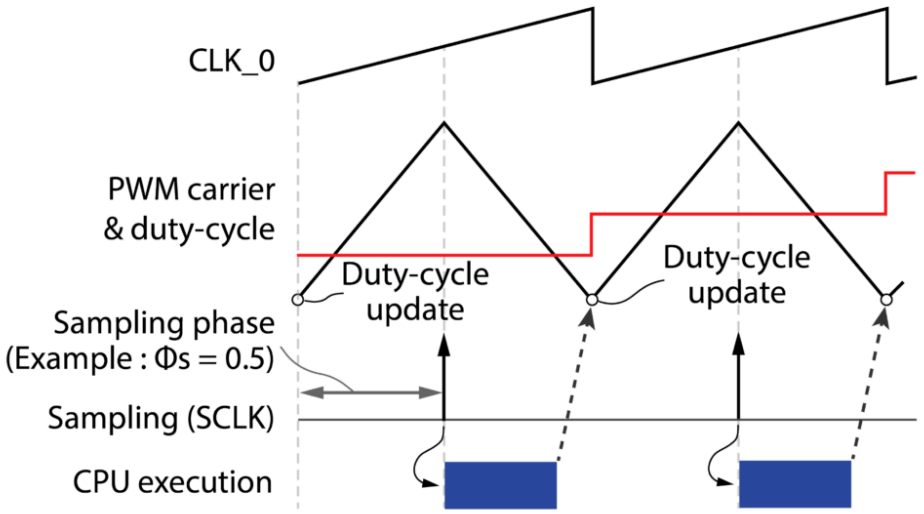

Figure 6 : Single rate update timing configuration

The remaining degree of freedom is the sampling phase, which must be selected according to the chosen sampling method (e.g., synchronous sampling, synchronous averaging), see sampling techniques. When relevant, the sampling phase needs to be adjusted to account for analog front-end delays. The practical selection of this sampling phase is discussed below.

Sampling phase configuration

On imperix controllers, configuring CLK0 (the sampling frequency) is only the first step. The sampling instant must also be defined by selecting a proper phase within the CLK0 period.

This phase is configured in the **CONFIG** block and is implemented by generating a dedicated sampling clock, SCLK, derived from CLK0. SCLK has the same frequency as CLK0, but with a constant phase shift. This shift sets the sampling instant within the PWM period and, immediately after the ADC acquisition delay, when the CPU control cycle starts. This sequence is shown in Fig. 7.

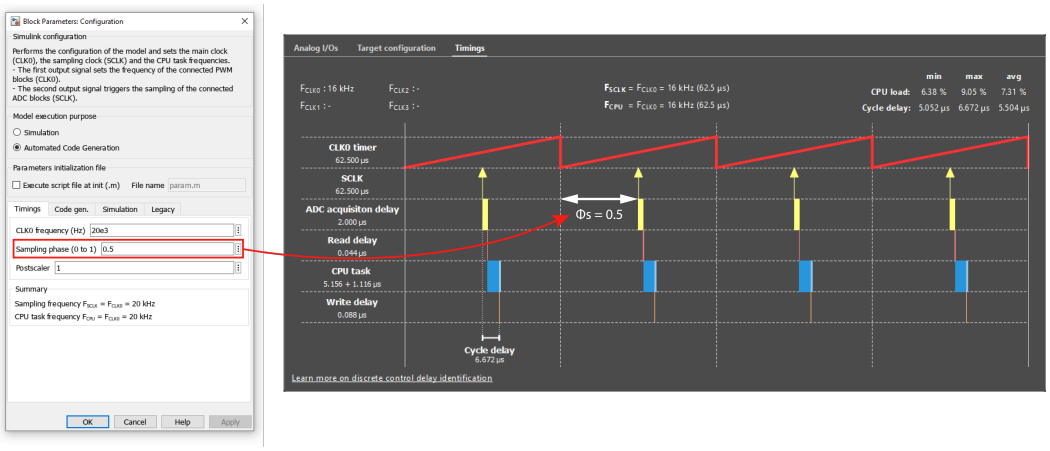

Figure 7 : Sampling phase configuration on imperix controller

On most applications, the required phase value depends largely on the sampling method.

- When synchronous averaging is used, the exact sampling instant within the PWM period is not critical because the measurement corresponds to a periodic average over a full switching period. In this scenario, the phase is typically adjusted to minimize the total control loop delay.
- When synchronous sampling is used, the sampling instant must be placed at a point that yields a representative ripple-averaged value (for example if a current is measured). This generally leads to two phase options, 0 or 0.5, which often have to be adjusted to compensate for the propagation delays inherent in the analog measurement chain, such as those resulting from low sensor bandwidth.

The sampling phase selection (0 or 0.5) is typically determined by the CPU execution time. A phase of 0.5 is generally preferred to minimize control delay, provided the CPU task completes within half a switching period. Alternatively, a phase of 0 is adopted for intensive computations to allow a full switching period for CPU execution.

Double rate update

In a double rate update control scheme on imperix controllers, ADC sampling and CPU execution remain tied to CLK0, while the PWM block(s) use a separate time base, typically CLK1 (but can also be CLK2 or CLK3). In order to obtain two duty cycles update opportunities within one PWM period, the PWM time base has to be set to $F_{\mathrm{CLK1}} = 0.5 \cdot F_{\mathrm{CLK0}}$. This configuration is illustrated in Fig. 8.

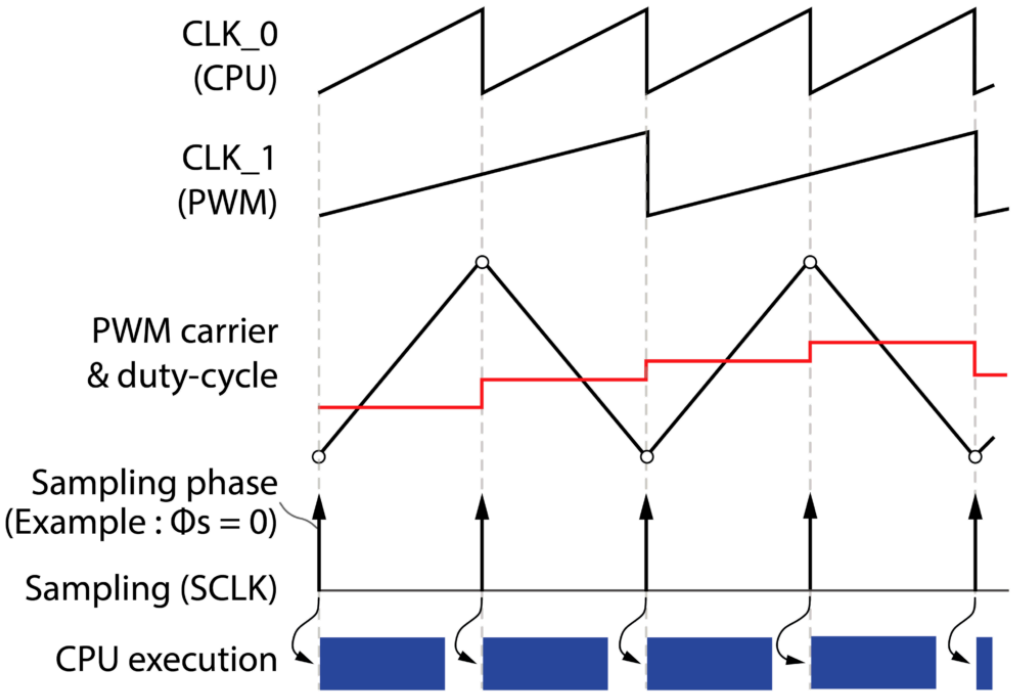

Figure 8 : Double rate update timing configuration on imperix controllers

Double rate update requires a double-edge (**triangular**) carrier as single-edge carriers (sawtooth) offer only one update instant per PWM period.

As with single-rate configurations, the optimal sampling phase in double-rate update mode depends on the chosen acquisition technique :

- When synchronous averaging is used, the acquisition must be configured to average over two periods of CLK0 (in the ADC block). Since CLK0 runs at twice the switching frequency in this configuration, this setting ensures that the average is calculated over one full PWM period.
- When synchronous sampling is used, the sampling phase is typically set to 0 to align the acquisition with representative ripple-averaged values. This value may be further adjusted to compensate for delays inherent in the analog measurement chain (such as limited sensor bandwidth).

From an implementation perspective, Fig. 9 summarizes the corresponding block-level structure and clock assignments for a double-rate update closed-loop buck

setup. Reference Simulink and PLECS models implementing this configuration are provided below and can be used as a starting point for application-specific designs.

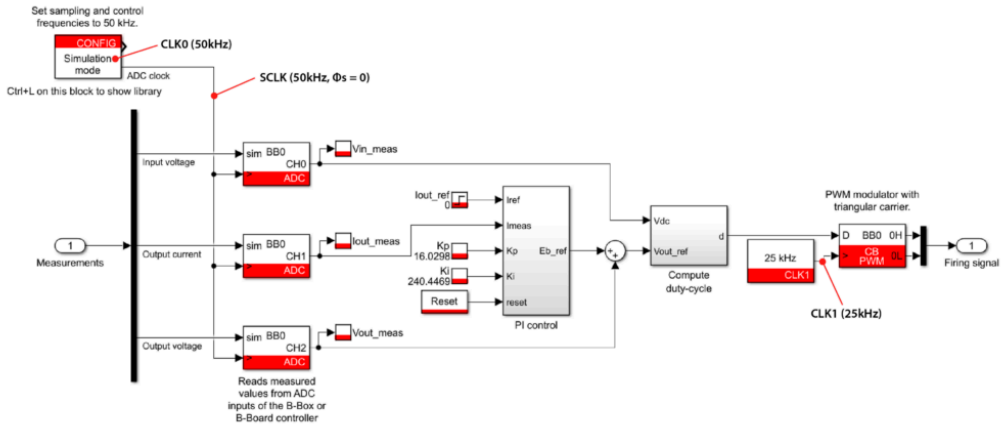

Figure 9: Double rate update configuration using Simulink

Simulink model

PN259_Double_rate_update_SimulinkDownload

PLECS model

PN259_Double_rate_update_PLECSDownload

Advanced timing configurations

Postscaler

Imperix controllers offer the flexibility to run the CPU at a decimated rate relative to CLK0. By adjusting the post-scaler in the CONFIG block, the CPU execution frequency is defined as $F_{\mathrm{CPU}} = F_{\mathrm{CLK0}}/Postscaler$.

While the CPU operates at this reduced rate, the sampling frequency remains synchronized with CLK0. This decoupling is particularly advantageous when controlling interleaved converters. In such topologies, the sampling phase must be specifically shifted for each converter leg to capture the current at the ideal moment. Consequently, the sampling frequency must be N times faster than the CPU execution rate, where N represents the number of legs.

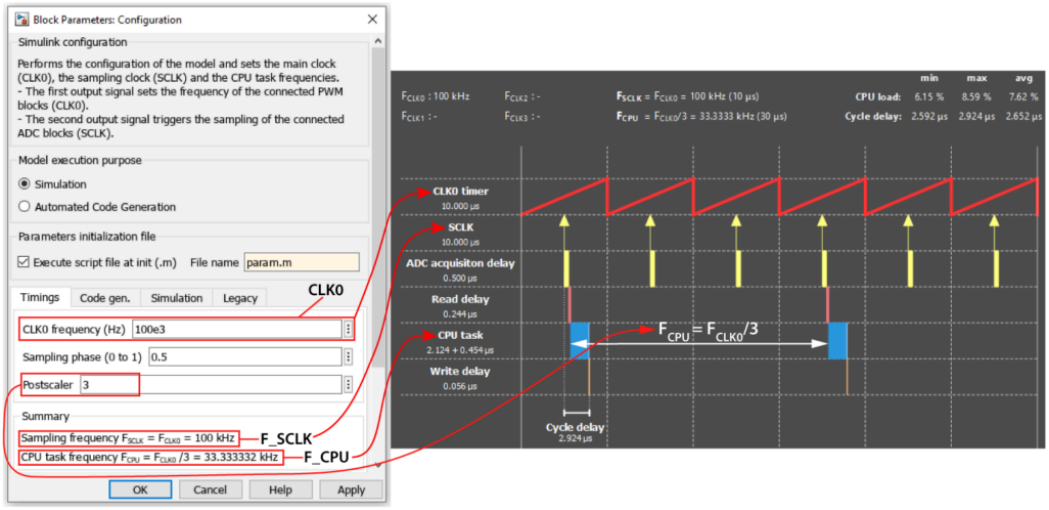

Figure 10: Postscaler configuration with N = 3 and corresponding timing view in Cockpit

Figure 10 illustrates the timing view within the Cockpit interface when a postscaler of 3 is applied. In this specific configuration, the CPU executes at a rate three times slower than both CLK0 and the sampling rate (SCLK).

Data history

Imperix controllers also offer the ability to retrieve sampling data captured between CPU interrupt interrupts through the 'data history' feature of the ADC block. This functionality is particularly advantageous when using a postscaler, as it enables the control algorithm to access the complete set of samples acquired during the preceding decimated interval. As illustrated in Fig.11, this ensures that all intermediate data points are available for processing, regardless of whether they were captured before the current CPU task was triggered.

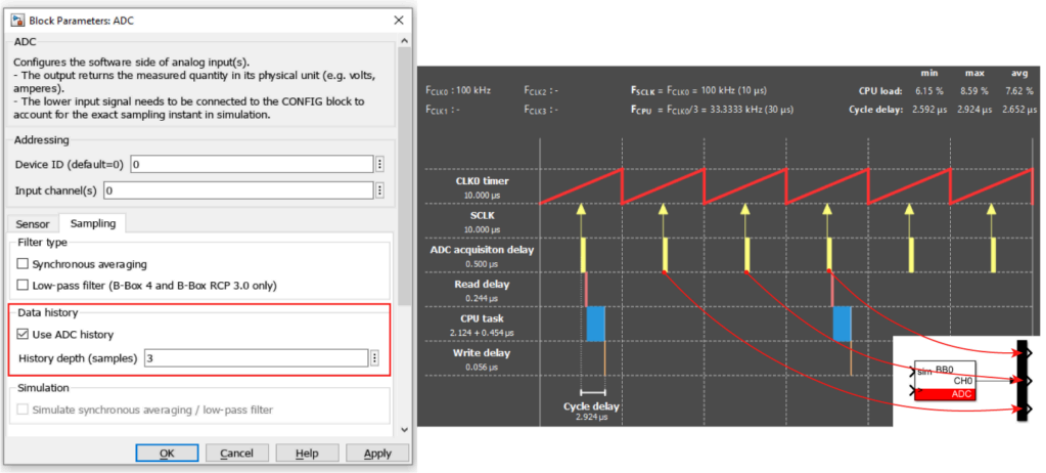

Figure 11: ADC history configuration with 3 samples

Variable switching frequency

While most converters operate at a fixed frequency, some applications require dynamic frequency adjustment for regulation. A primary example is the LLC Resonant Converter, where the output voltage is controlled by modulating the switching frequency.

On imperix controllers, this is achieved by configuring the CLK1-3 blocks to allow for real-time frequency updates, as shown in Fig.12. See also Variable frequency operation with imperix controllers.

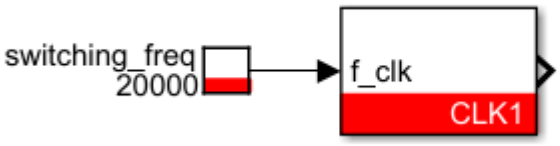

Figure 12 : Variable frequency clock using Simulink

When using variable switching frequencies, the sampling frequency remains fixed and tied to CLK0. Consequently, any form of synchronous acquisition is not possible, as the sampling instants will no longer align with the varying PWM periods.

Multi-rate execution

Both Simulink and PLECS provide the flexibility to execute specific parts of a control algorithm at a lower frequency than the main execution rate (F_{CPU}). As this is managed via dedicated software blocks, it requires no modifications to the controller's timing configurations. This function is described in more detail in PN145 – Multi-rate control with Simulink and PN155 – Multi-rate control with PLECS.

This feature may be useful for cascaded control strategies, where high-level outer loops, such as voltage or motor speed regulation, can run at a lower rate, while the inner loops continue to operate at the main execution rate for better dynamic performance. Example of multi-rate execution can be found in Motor speed control or in MPPT algorithms.

Related topics

PN258 – Sampling techniques for power electronics for the various sampling patterns and associated delays.
PN108 – Analog I/O configuration for imperix controllers explains how to configure (via software) the various sampling schemes
PN135 – Simulation essentials with Simulink explains how to use and configure the ACG SDK blocks (for Simulink)
PN137 – Simulation essentials with PLECS explains how to use and configure the ACG SDK blocks (for PLECS)
PN121 – Variable frequency operation with imperix controllers describes the variable switching frequency configuration.
RealSync technology explains how synchronisation between controllers is achieved.